Generic Binding Signatures (Algebra, Languages and Computation)

AUTHOR(S):

Tanaka, Miki

CITATION:

Tanaka, Miki. Generic Binding Signatures (Algebra, Languages and Computation). 数理解析研究所講究録 2005, 1437: 37-44

ISSUE DATE:

2005-06

URL:

http://hdl.handle.net/2433/47492

数理解析研究所講究録 1437 巻 2005 年 37-44

Generic Binding Signatures

Miki Tanaka
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi, Koganei, Tokyo, Japan, 184-8795
Tel:+81-(0)42-327-5782, Email:miki.tanaka@nict.go.jp

Abstract

We present a new definition of binding signatures, i.e., signatures for operators with variable binding, along the line of the work by Fiore, Plotkin and Turi. The definition given by them was only for cartesian binders, as found in λ-calculus, whereas ours cover binding in any contexts generated by a pseudo-monad S on Cat. Our generalisation also includes the construction of initial algebra semantics with substitution for the signatures.

1 Introduction

In the study of programming languages, the notion of variable binding has always been a common one, yet presenting subtle and difficult problems. As early as in late 60's, de Bruijn discussed in [1] how to dispense with names of bound variables, and started AUTOMATH, a project for automated verification of mathematics. The basic idea of de Bruijn was to provide a systematic way of describing a canonical representative of equivalence classes induced by renaming of bound variables in a formal system, i.e., α-equivalence classes.

In 1999, in LICS, Fiore, Plotkin and Turi [4] then gave a model for such α-equivalence classes on a presheaf category, together with the definition of signatures for binding operators. The model has a natural operation of substitution on it, which is compatible with algebras for the signatures. Their construction was applied only to binding in cartesian contexts as found in λ calculus, although underneath their work lied some constructions which generalise to a framework that accounts for variable binding and substitution in more general settings. This fact was also endorsed by the paper [16] on linear binders, which gave constructions that go parallel to those in [4] in a slightly different setting. The definitions of binding signatures given in [4] and [16], for cartesian and linear contexts, respectively, coincide. The difference becomes visible only when one tries to construct algebras for the signatures in different categories.

However, when one wants to deal with more complex settings, one immediately sees that their definition of binding signatures is not good enough. As a leading example of such cases we focus on bunched contexts [15], where contexts

are freely generated from both cartesian and linear variables. In this case, one needs to be able to specify for each operator how it combines cartesian and linear variables, and how many variables it should bind (including none.) Our definition of binding signatures enables this for any given context generated by a pseudo-monad S on Cat.

The paper is organised as follows: in Section 2 we give a list of pseudo-monads that generate the categories of contexts we use in later sections. Then, after recalling the definitions of binding signatures for cartesian and linear cases in Section 3.1, we give the definition of binding signatures for generic contexts in Section 3.3. The following section (Section 4) contains the discussion on construction of algebras for the binding signatures, leading to the initial algebra semantics. In Section 5, we look at the leading example of bunched contexts, which is explained in detail alongside the comparison with the cartesian and linear cases.

In this paper, our emphasis is on the definitions of binding signatures. For readers interested in the semantic aspects of variable binding for generic contexts are referred to [17, 12]; in particular, the mathematical foundations of how pseudo-distributivity plays an important role in the whole construction can be found in [17]. For the further analysis and proofs related to signatures, see [13]. The constructions in the paper are based on the category theory. In particular, in order to have a full understanding of the details, some knowledge on pseudo-monads is essential, which is not presented in this paper. For readers who are not familiar with such techniques are referred to other documents such as [17, 12].

Notation 1.1. We sometimes use a natural number k to denote the discrete category with k objects, expressed as $\{1, \ldots, k\}$. It should be clear from the context whether k denotes a natural number or a category, although the reader needs to be aware of the obvious overloading.

2 Categories for Contexts

The main ingredient of the discussion in this paper is contexts; in [4], the contexts they focused on were cartesian ones, and they used the category $\mathbb{F}$ of finite sets and all functions to model them. In [16], the contexts dealt there were linear ones, modelled by the category $\mathbb{P}$ of finite sets and permutations between them. We take the view that contexts are generated by suitable pseudo-monads on Cat, the category of small categories. The notion of pseudo-monad on Cat is a variant of the notion of monad on Cat. For space reasons, we shall not define pseudo-monads, their 2-categories of pseudo-algebras, etcetera, here, beyond remarking that they are the definitive variant of the notions of monad, algebra, etcetera, that respect natural transformations and for which equalities in the various axioms are systematically replaced by coherent isomorphisms [12, 17].

In the following, we give descriptions of several pseudo-monads and the contexts generated by them. The first two examples are those implicitly used in the papers [4] and [16]. The third is for our leading example, which we will investigate further in Section 5.

Example 2.1. Let T_{fp} denote the pseudo-monad on *Cat* for small categories with finite products. The 2-category *Ps-T_{fp}-Alg* has objects given by small categories with finite products, maps given by functors that preserve finite products in the usual sense, i.e., up to coherent isomorphism, and 2-cells given by all natural transformations. So *Ps-T_{fp}-Alg* is the 2-category *FP*. The category $T_{fp}(X)$ is the free category with finite products on X. Taking $X = 1$, the category $T_{fp}(X)$ is given, up to equivalence, by the category of cartesian contexts $\mathbb{F}^{op}$ used by Fiore et al [4].

Example 2.2. Let T_{sm} denote the pseudo-monad on *Cat* for small symmetric monoidal categories. The 2-category *Ps-T_{sm}-Alg* has objects given by small symmetric monoidal categories, maps given by strong symmetric monoidal functors, i.e., functors together with data and axioms to the effect that the symmetric monoidal structure is preserved up to coherent isomorphism, and 2-cells given by all symmetric monoidal natural transformations, i.e., those natural transformations that respect the symmetric monoidal structure. Therefore, *Ps-T_{sm}-Alg* is the 2-category $SymMon_{str}$ and $T_{sm}(X)$ is the free symmetric monoidal category on X. Taking $X = 1$, it follows, up to equivalence, that $T_{sm}(X)$ is the category $\mathbb{P}^{op}$ of finite sets and permutations used by Tanaka [16] for modelling linear contexts.

Example 2.3. Combining the first two examples by taking the sum of pseudo-monads, we may consider the pseudo-monad T_{BI} on *Cat* for small symmetric monoidal categories with finite products. The 2-category *Ps-T_{BI}-Alg* has objects given by small symmetric monoidal categories with finite products, maps given by strong symmetric monoidal functors that preserve finite products, and 2-cells given by all symmetric monoidal natural transformations. This structure is the free category on 1 independently generated by finite-product and symmetric monoidal structures used in the Logic of Bunched Implications [15]. The objects of $T_{BI}(X)$ where $X = 1$ are precisely the *bunches* of Bunched Implications. More syntactic descriptions of bunched contexts are found in later sections.

3 Binding Signatures

In [4] and [16], their definitions of binding signatures, which are identical, are given as a generalisation of the usual notion of signatures in universal algebra. Instead of having simple natural numbers as arities, they use sequences of natural numbers to account for the number of variables to be bound in each argument of an operator. Unfortunately, this generalisation is not general enough when one wants to give signatures for more complex contexts, such as those for the Logic of Bunched Implications. By further generalising the definition in a category theoretic way, we give a new definition of binding signatures which can describe such cases.

The definition of binding signatures is followed by the construction of a corresponding endofunctor on a suitable category. In [4] and [16], the presheaf

categories $[\mathbb{F}, Set]$ and $[\mathbb{P}, Set]$ are used respectively to define the endofunctors and then to construct algebras. For the case of the generic contexts generated by S, we use the category $[(S1)^{op}, Set]$ of presheaves.

We first look at the cases where $S = T_{fp}$ and $S = T_{sm}$, the pseudo-monads for cartesian contexts and linear contexts. Then we give the definition of binding signatures for S in general.

3.1 Cartesian and Linear Cases

As mentioned earlier, Fiore, Plotkin and Turi in their paper in LICS'99 [4] discussed the case $S = T_{fp}$, for carteisan contexts. The contexts are modelled by the category $\mathbb{F}^{op}$, which is equivalent to $T_{fp}1$ (Example 2.1). For linear contexts, the case $S = T_{sm}$ in [16] uses the category $\mathbb{P}^{op}(= \mathbb{P})$, which is equivalent to $T_{sm}1$ (Example 2.2). Therefore, they use different categories for constructing algebras, although their definitions of binding signatures coincide.

Definition 3.1 ([4, 16]). A *binding signature* $\Sigma = (O, a)$ consists of a set O of operations, and a function $a : O \to \mathbb{N}^*$. Elements of the codomain of a are called *arities*.

If $a(o) = \langle n_1, \ldots, n_k \rangle$ for an operator o, it means that o takes k arguments and binds n_i variable in the i-th argument.

Example 3.2. The signature Σ_λ of λ-calculus

$$t ::= x \mid \lambda x.t \mid \mathsf{app}(t, t)$$

is given by $\Sigma_\lambda = (\{\lambda, \mathsf{app}\}, a)$. The arities of operators are given as $a(\lambda) = \langle 1 \rangle$ and $a(\mathsf{app}) = \langle 0, 0 \rangle$.

Given a signature, one constructs an endofunctor associated to it on a suitable category in which one considers algebras of the signature. For the cartesian case, an endofunctor associated to a binding signature Σ is constructed on the category $[\mathbb{F}, Set]$. The operation on carteisan contexts corresponds to the operation $+$ (coproduct in $\mathbb{F}$) in the category $\mathbb{F}$. This operation naturally extends to the operation on terms, in this case the operation $\times$ of taking products in $[\mathbb{F}, Set]$. Using these two operations, the endofunctor Σ on $[\mathbb{F}, Set]$ is defined to send X to

$$\Sigma X = \coprod_{\substack{o \in O \\ a(o) = \langle n_i \rangle_{1 \le i \le k}}} X(n_1 + -) \times \cdots \times X(n_k + -).$$

Similarly, but instead of $+$ and $\times$, operations $\otimes$ and $\overline{\otimes}$ in $\mathbb{P}$ and $[\mathbb{P}, Set]$, respectively, are used for the linear contexts: the endofunctor Σ on $[\mathbb{P}, Set]$ is defined to send X to

$$\Sigma X = \coprod_{\substack{o \in O \\ a(o) = \langle n_i \rangle_{1 \le i \le k}}} X(n_1 \otimes -) \overline{\otimes} \cdots \overline{\otimes} X(n_k \otimes -).$$

3.2 Bunched contexts

Naturally, we can think of more variations in contexts other than cartesian and linear cases. One of such structural variations is the contexts found in the Logic of Bunched Implications, a logic for resources and sharing introduced by Pym in 1999 [15]. In this logic, the contexts are called "Bunches", and two different kinds of operations are allowed on them; one is the usual operation in cartesian contexts, and the other is that in linear contexts. Hence the category of bunched contexts, denoted by $T_{BI}1$ as in Example 2.3, is given as a symmetric monoidal category with finite products, freely generated on 1. The two operations on contexts result in two different operations on terms, as seen in the $\alpha\lambda$-calculus [10], which corresponds to the logic:

$$t ::= x \mid \lambda x.t \mid \mathsf{app}(t_1, t_2) \mid \alpha x.t \mid @(t_1, t_2)$$

where λ and app are the operators of linear binding and application, while α and @ are those for cartesian binding and application.

One can immediately see that it is impossible to give signatures for such a calculus with the definition of binding signatures given above. So, We need something more sophisticated than the two binding signatures defined so far.

3.3 Binding Signatures for Context S

As we have seen in the preceding sections, in order to give the signature for $\alpha\lambda$-calculus, the definitions of binding signatures given in [4, 16] are not broad enough. In particular, it is necessary to be able to specify, firstly, how one wants to apply arguments to an operator, and secondly, how one wants to bind variables in each of the arguments. The definition we give below [13] provides the facility for these issues.

Definition 3.3 (Generalised, [13]). For a pseudo-monad S on Cat, a *binding signature* $\Sigma^S = (O, a)$ is a set of operations O together with an arity function $a : O \to Ar_S$ where an element $(k, \alpha, (\alpha_i)_{1 \leq i \leq k})$ of Ar_S consists of a natural number k, an object α of the category Sk, and, for $1 \leq i \leq k$, an object α_i of the category $S2$.

The idea is that, if an operator o has an arity $(k, \alpha, (\alpha_i)_{1 \leq i \leq k})$, then it takes k arguments, which are combined in a way specified by α, and in each of the k arguments binds variables in the way specified by α_i, for $i = 1, \ldots, k$.

4 Algebras for the Signatures

In this section, we present the generic construction of the endofunctor associated to a generic binding signature Σ^S, on the presheaf category $[(S1)^{op}, Set]$. For this purpose, we need to explain how objects of Sk, where k is a discrete category, gives rise to a functor. Then, for each operator o, we apply this to the components of its arity to obtain an endofunctor which interprets o. Putting

all together by taking the coproduct over all the operators, we obtain an endofunctor associated to the signature.

4.1 Functors induced by objects of Sk

In general, given an S-algebra (A, a), each object of Sk induces a functor from A^k to A. Let γ be an object of Sk. Then γ induces a functor $\overline{\gamma}_A : A^k \to A$ by composing arrows as:

$$A^k \cong A^k \times 1 \xrightarrow{S\times\gamma} (SA)^{Sk} \times Sk \xrightarrow{ev_\gamma} SA \xrightarrow{a} A.$$

So, given an object f of A^k, the value $\overline{\gamma}_A(f)$ is $a(Sf(\gamma))$ as seen in:

$$\begin{array}{ccccccc} A^k & \cong A^k \times 1 & \xrightarrow{S\times\gamma} & (SA)^{Sk}\times Sk & \xrightarrow{ev_\gamma} & SA & \xrightarrow{a} & A \\ f & (f,1) & & (Sf,\gamma) & & Sf(\gamma) & & a(Sf(\gamma)) \end{array}.$$

Since f can be expressed as a k-tuple $(s_1, \ldots, s_k)$ of objects of Sk, the value $Sf(\gamma)$ is calculated by substituting s_i for all i's appearing in γ, for $1 \leq i \leq k$.

For the case of contexts generated by S, using $\star$ to denote an operation in Sk, and $\overline{\star}$ for its extension to the presheaf category, we have the following constructions: since $S1$ is a free category, it has a natural structure of S-algebra $(S1, \mu)$. So $\alpha_i \in S2$ induces a functor

$$\overline{\alpha_i}_{S1} : (S1)^2 \to S1$$

For example, the object 1 in $S2$ induces a functor $\overline{1}_{S1} : (S1)^2 \to S1$ such that (s_1, s_2) in $S2$ is sent to s_1 And, the object $1 \star 2$ in $S2$ induces a functor $\overline{1 \star 2}_{S1} : (S1)^2 \to S1$ such that (s_1, s_2) in $S2$ is sent to $s_1 \star s_2$.

On the other hand, the category $[(S1)^{op}, Set]$ is also an S-algebra. So α in Sk induces a functor

$$\overline{\alpha}_{[(S1)^{op},Set]} : [(S1)^{op}, Set]^k \to [(S1)^{op}, Set].$$

Letting $\alpha = 1 \star 2$ in $S2$, $1 \star 2$ induces a functor that sends $(X, Y) \in S2$ to $X \overline{\star} Y$

$$\overline{1 \star 2}_{[(S1)^{op},Set]} : [(S1)^{op}, Set]^2 \to [(S1)^{op}, Set].$$

Example 4.1. For $1 \otimes 2$ in $T_{sm}2$, the value at $f = (m, n)$ in $\mathbb{P}^2$ is

$$\overline{1 \otimes 2}_{\mathbb{P}}(f) = m \otimes n$$

In the next section, using the above construction, we define an endofunctor that corresponds to a generic binding signature Σ^S.

4.2 Σ^S-algebras

Using the functors defined as in the previous subsection, we now construct an endofunctor for a given binding signature Σ^S for contexts S. Each operator

o with an arity $(k, \alpha, (\alpha_i)_{1 \leq i \leq k})$ induces an endofunctor on $[(S1)^{op}, Set]$ that sends X to

$$\overline{\alpha}_{[(S1)^{op}, Set]}(X(\overline{\alpha_1}\ _{S1}(1, -)), \ldots, X(\overline{\alpha_k}\ _{S1}(1, -))).$$

Then, a signature Σ^S induces an endofunctor Σ^S on $[(S1)^{op}, Set]$ that sends X to

$$\coprod_{\substack{o \in O \\ a(o) = (k, \alpha, (\alpha_i)_{1 \leq i \leq k})}} \overline{\alpha}_{[(S1)^{op}, Set]}(X(\overline{\alpha_1}\ _{S1}(1, -)), \ldots, X(\overline{\alpha_k}\ _{S1}(1, -)))$$

by taking coproducts over all operators in O. One has initial algebra semantics with substitution for a signature Σ^S in $[(S1)^{op}, Set]$ ([17, 12, 14]).

5 Bunched Contexts

In this section, as one example demonstrating that our definition of binding signatures is indeed a generalisation of those found in [4, 16], we show the signature of the $\alpha\lambda$-calculus, whose syntax was given in Section 5. We use the category $T_{BI}1$ as the model of contexts, and define an endofunctor on $[(T_{BI})^{op}, Set]$.

In order to define the signature $\Sigma_{\alpha\lambda}$ for this calculus, we choose k, $\alpha \in T_{BI}k$ and $\alpha_i \in T_{BI}2$, for $1 \leq i \leq k$, for each of the four operators. For instance, the linear binder λ takes one argument, hence k should be 1, and since there is no way of combining single argument, α should be the object 1 of $T_{BI}1$ (this object induces the identity functor). Finally, since λ binds one variable linearly, α_1 is given by the object $1 \otimes 2$ in $T_{BI}2$. To summarise, the arities for the operators of $\alpha\lambda$-calculus are given as follows:

$$a(\lambda) = (1, 1, 1 \otimes 2) \qquad a(\mathsf{app}) = (2, 1 \otimes 2, (2, 2))$$
$$a(\alpha) = (1, 1, 1 \times 2) \qquad a(@) = (2, 1 \times 2, (2, 2))$$

Then, an easy calculation shows that this signature $\Sigma_{\alpha\lambda}$ induces an endofunctor on $[(T_{BI})^{op}, Set]$ that sends X to

$$\Sigma_{\alpha\lambda}X = X(1 \otimes -) + X \otimes X + X(1 + -) + X \times X.$$

6 Conclusion

We have presented a definition of binding signatures for contexts generated by a pseudo-monad S on Cat, generalising the definitions given in [4] and [16]. Our definition accommodates contexts such as those for the Logic of Bunched Implications, which was not the case for the definitions in [4, 16]. The endofunctor on $[(S1)^{op}, Set]$ which the signature induces is constructed using the fact that, given an S-algebra (A, a), each object of Sk induces a functor from A^k to A. One has initial algebra semantics for the signature in this presheaf category, which also has a structure that models substitution [17, 12, 14].

References

[1] N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. Indagationes Mathematicae, 34:381–392, 1972.

[2] B. Day. On closed categories of functors. In *Lecture Notes in Mathematics 137*, pages 1–38. Springer-Verlag, 1970.

[3] M. Fiore, Semantic analysis of normalisation by evaluation for typed lambda calculus. In *Proc. PPDP 02*, ACM Press, pages 26–37, 2002.

[4] M. Fiore, G. Plotkin, and D. Turi. Abstract syntax and variable binding. In *Proc. LICS 99*, pages 193–202. IEEE Press, 1999.

[5] M. Gabbay and A. M. Pitts. A new approach to abstract syntax involving binders. In *Proc. LICS 99*, pages 214–224. IEEE Press, 1999.

[6] M. Hofmann, Semantical analysis of higher-order abstract syntax. In *Proc. LICS 99*, IEEE Press, pages 204–213, 1999.

[7] G. M. Kelly. *Basic Concepts of Enriched Category Theory, London Math. Soc. Lecture Notes Series 64* Cambridge University Press, 1982.

[8] S. Mac Lane. *Categories for the Working Mathematician.* Springer-Verlag, 1971.

[9] M. Miculan and I. Scagnetto, A framework for typed HOAS and semantics. In *Proc. PPDP 2003, ACM Press*, pages 184–194, 2003.

[10] P. O'Hearn, *On Bunched Typing, Journal of functional Programming*, 13:747-796, Cambridge University Press, 2003.

[11] A. J. Power. A Unified Category-Theoretic Approach to Variable Binding. In *Proc. MERLIN 2003, ACM Digital Library*, 2003.

[12] A. J. Power and M. Tanaka. Pseudo-Distributive Laws and Axiomatics for Variable Binding. Submitted.

[13] A. J. Power and M. Tanaka. Binding Signatures for Generic Contexts In *Proc. TLCA 2005, LNCS 3461*, pages 308–323, 2005.

[14] A. J. Power and M. Tanaka. A Unified Category-Theoretic Semantics for Binding Signatures in Substructural Logics. Submitted.

[15] D. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications, Applied Logic Series.* Kluwer, 2002.

[16] M. Tanaka. Abstract syntax and variable binding for linear binders. In *Proc. MFCS 2000, LNCS 1893*, pages 670–679, 2000.

[17] M. Tanaka. Pseudo-Distributive Laws and a Unified Framework for Variable Binding. Edinburgh Ph.D. thesis, 2004.